VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra Informatiky

Silverlight Dynamic Languages

Silverlight Dynamic Languages

2012 Daniel Šudřich

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Daniel Šudřich**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Silverlight Dynamic Languages

Silverlight Dynamic Languages

Zásady pro vypracování:

Silverlight Dynamic Languages SDK dovoluje vývojářům použití dynamických jazyků běžících na Dynamic Language Runtime (DLR) k tvorbě Silverlight aplikací. DLR umožňuje dynamickým jazykům (JScript, IronPython) součinnost s .NET Framework. Obsahuje také nástroje k tvorbě nových jazyků. Cílem této práce bude provést výkonnostní testy s technologií Silverlight Dynamic Languages SDK a zhodnotit výsledky vykonaných experimentů.
Jednotlivé body práce jsou následující:
1. Prostudovat technologií Silverlight.
2. Prostudovat a popsat Silverlight Dynamic Languages SDK.
3. Tvorba ukázkových příkladu využití Silverlight Dynamic Languages SDK.
4. Provést výkonostní testy s technologií Silverlight Dynamic Languages SDK.
5. Zhodnocení výsledků experimentů.

Seznam doporučené odborné literatury:

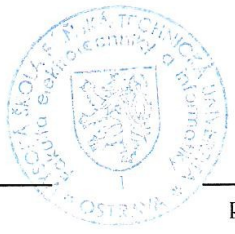
Matthew MacDonald: Pro Silverlight 4 in C#, 2010. 912 s. ISBN-10: 1430229799. ISBN-13: 978-1430229797.
Michael J. Foord: IronPython in Action, 2009. 480 s. ISBN-10: 1933988339. ISBN-13: 978-1933988337.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Peter Scherer**

Datum zadání: 19.11.2010

Datum odevzdání: 04.05.2012



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal."

V Ostravě dne 4.5.2012 .....

Chtěl bych touto formou poděkovat svému vedoucímu práce, kterým byl Ing. Peter Scherer. Bez jeho vize, nápadů a odborných rad by tato práce nikdy nedosáhla zdárného svého konce.

Abstrakt

Silverlight Dynamic Languages SDK Runtime umožňuje vývojářům psát Silverlight aplikace pomocí dynamických jazyků, kterými jsou IronPython a IronRuby. Silverlight je aplikační platforma vytvořená společností Microsoft, která je určena pro vývoj nové generace RIA a multimediálních aplikací. Tato práce postupně představuje .NET Framework, dál rozebírá Dynamic Language Runtime, kde hlavním bodem je představení jeho architektury. Následně představuje Silverlight, její historii a možnou budoucnost. Poté je čtenáři poskytnut podrobný a velice přehledný návod jak vytvořit své první Silverlight aplikace pomocí dynamických jazyků se spojením prezentační vrstvou reprezentovanou pomocí XAML. Na závěr jsou v práci provedeny výkonnostní testy s dynamickými jazyky, které zkoumají rychlost těchto jazyků.

Klíčová slova: Dynamic Language Runtime, Silverlight, Silverlight Dynamic Language Runtime, IronPython, IronRuby

Abstract

Silverlight Dynamic Language Runtime allows developers to write Silverlight applications using dynamic languages, which are IronPython and IronRuby. Silverlight is an application platform created by Microsoft, that is designed to develop new generation RIA and multimedia applications. This work is gradually shows .NET Framework, further analyzes the Dynamic Language Runtime, where the main point is the performance of its architecture. Following is Silverlight, its history and possible future. Then is given the reader a detailed and very clear instructions on how to create your first Silverlight applications using dynamic languages with represented by combining presentation layer using XAML. At the end, the work executed performance tests with dynamic languages, which examine the speed of these languages.

Klíčová slova: Dynamic Language Runtime, Silverlight, Silverlight Dynamic Language Runtime, IronPython, IronRuby

Seznam použitých zkratek a symbolů:

CLR – Common Language Runtime
DLR – Dynamic Language Runtime
DOM – Document Object Model
HTML – HyperText Markup Language
JIT – Just In Time
JSON - JavaScript Object Notation
RIA – Rich Interner Applicaiton
SDK – Software development kit
SQL – Structured Querrz Language
VS – Visual Studio
WCF – Windows Communication foundation
WPF – Windows Presentation Foundation
XAML – Extensible Application Markup Language
XML – Extensible Markup Language

Obsah

1. Úvod

Pod pojmem programovací jazyk rozumíme prostředek pro zápis programů, jež mohou být provedeny na počítači. V tomto smyslu je programovací jazyk komunikačním nástrojem mezi uživatelem počítače, který jeho jazykovými prostředky specifikuje algoritmus řešení daného problému, a počítačem, jenž svými technickými prostředky algoritmus interpretuje a realizuje tak transformaci vstupních údajů na výstupní.

První programovací jazyky existovaly ještě dříve než počítače. Už v 19. století se používaly programovatelné tkalcovské stavy nebo perforované papírové válce pro samohrající piána, které implementovaly to, co bychom dnes rozeznali jako vzory doménově orientovaných programovacích jazyků. Začátkem dvacátého století se již data uchovávala pomocí děrných štítků. [11]

V prvním období existence počítačů zastával roli programovacího jazyka výhradně strojový kód, nazývaný proto také strojový jazyk. Vyjadřovacími prostředky strojového jazyka jsou instrukce. Jejich sémantika je definována technickými prvky počítače. Z této skutečnosti vyplývá základní nedostatek programování ve strojovém jazyce - závislost na konkrétním typu počítače. Dalšími, z hlediska programování nepříznivými, rysy strojového jazyka je velmi nízká úroveň popisovaných akcí a pro člověka nepohodlný číselný zápis instrukcí.

Dnešní doba vypadá mnohem jinak. Přes kompilované, interpretované a neprocedurální jazyky největší obliby dosáhly objektově orientované programovací jazyky, které řadíme do skupiny procedurálních. Některé programovací jazyky (např. C++, Python, Object Pascal) umožňují programátorovi kombinovat různé přístupy. Část řešení může být například vyjádřena zápisem funkcí a procedur (strukturované programování), část řešení může využívat čistě objektový přístup.

V dnešní moderní době, kdy internet hýbe celým světem, jsou dynamické webové stránky a kompletně webové prezentace jedním z nejdůležitějších šancí úspěchu pro firmy a také živnostníky. Firmy se na internetu prezentují stále více, prakticky firma, která není na internetu, nemůže být úspěšná. Samozřejmě internet není jen o využití obchodník příležitostí, ale také o zábavě. Internet je prostě fenomén. Ovšem nikdy by se nestal tak důležitou a rozšířenou částí našich životů, pokud by zůstal pouze ve spojení HTML a CSS. Příchod JavaScriptu ovlivnil hodně věcí a postupně se objevovali další možností rozšíření webu. Jedním z nejúspěšnějších rozšíření webové prezentace se nakonec stal plug-in firmy Adobe s názvem Flash Player. Ovšem jedna z hlavních firem na poli informatiky řekla dost a tým z Microsoftu v roce 2007 vyrazil do boje proti Adobe Flash Playeru právě s novou technologií Silverlight.

Cílem bakalářské práce bylo prostudovat a popsat technologii Silverlight a její vylepšení v podobě Silverlight Dynamic Languages SDK, které umožňuje psát Silverlight aplikace pomocí dynamických jazyků. Následně bylo cílem páce tvorba ukázkových příkladů s využitím Silverlight Dynamic Languages, provedení a zhodnocení výkonnostních testů.

Bakalářská práce postupně představuje dynamické jazyky, .NET Framework a modul Dynamic Language Runtime(DLR), který je stěžejním pro dynamické jazyky a přidání dynamických vlastnosti statickým jazykům.

Ve třetí kapitole je představena technologie Silverlight, která je určena pro vývoj RIA aplikací nové generace. Kapitola obsahuje popis Silverlight, její historii a také poskytuje náhled na její budoucnost.
Další kapitola se už podrobně zabývá balíkem Silverlight Dynamic Languages SDK, pomocí kterého je možné vytvářet Silverlight aplikace za pomocí dynamických jazyků, kterými jsou v tomto případě IronPython a IronRuby. Silverlight Dynamic Languages SDK původně umožňovalo psát Silverlight aplikace také pomocí třetího jazyka a jím byl Managed JScript. Tento jazyk byl od SDK verze 0.5

stažen, protože Microsoft zastavil vývoj. Díky tomuto rozhodnutí se práce nebude věnovat tomuto jazyku.

Pátá kapitola poskytuje čtenáři podrobný a jasný návod pro tvorbu prvních Silverlight aplikací pomocí dynamických programovacích jazyků. V šesté kapitola bude provedeno několik výkonnostních testů se Silverlight jazyky IronPython a IronRuby.

2. Dynamické jazyky v .NET

2. 1 Dynamické jazyky

Proč vlastně sáhnout k využití dynamických programovacích jazyků ? Proč přidat dynamické vlastnosti stávajícím statickým jazykům ? Tohle jsou dvě zásadní otázky, na které vlastně ani nejde nalézt odpovědi. To všechno díky nesmiřitelným dvěma táborům odborníků, kteří se už léta dohadují, kde je vlastně pravda. Tato část přiblíží obvyklé důvody, kterými se ohánějí lidé preferující dynamické jazyky.

Dynamické jazyky nabírají znovu na síle a to všechno díky vývoji webových aplikací. Dynamické jazyky byly nejpoužívanější v 80. letech pro skriptování webu a právě složitější a interaktivnější webové prezentace jsou důvodem jejich návratu na scénu. Také se však více dostávají do popředí oblíbenosti statické jazyky doplněné o dynamické vlastnosti pro jejich větší produktivitu při vývoji webových aplikací. {1}

Dynamické jazyky jsou flexibilnější než jazyky statické. Programování jejich využití je méně restriktivní a rychlejší. Největší výhodou dynamických jazyků je dynamická kontrola, která probíhá za běhu programu. Jazyky používající dynamické typování nepožadují specifikaci datového typu u proměnných a ty mohou tudíž odkazovat na hodnotu jakéhokoli typu. Mezi hlavní dynamické jazyky patří například Python, Ruby, Smalltalk a v České republice velice rozšířený jazyk PHP. Dynamické typování je pružnější, protože programy generují typy a funkcionalitu na základě běhových dat, avšak díky tomuto může docházet k častějším běhovým chybám programu, kdy hodnota nabude neočekávaného typu a pokračuje v dalším zpracování. Běhová chyba se projeví až daleko od místa svého vzniku a je těžko odhalitelná. [12]

Existuje také kombinace statického a dynamického typování, protože přítomnost statického typování v programovacím jazyce nemusí nutně znamenat absenci všech mechanismů dynamické typové kontroly. Programovací jazyk Java a další jazyky používají sice statické typování, ale pro některé operace požadují testování za běhu. .NET Framework 4.0 dokonce podporuje variantu dynamického typování pomocí namespace Systém.Dynamic, který je stále statický, ale jeho možnosti jsou zkoumány až za běhu. Všechny jazyky používané v rámci platformy .NET (C++, C#, J#, Visual Basic) se tak stávají plně dynamickými. Každý objekt může být implicitně přetypován do typu dynamic. Nad dynamickým typem je možné provádět všechny operace jako volání metod, přístup k prvkům a vlastnostem, použití indexem i použití jako volání delegáta. [12]

Komunita okolo dynamických jazyků je velice silná. Tito programátoři jsou velice věrní svým jazykům a plně věří v jejich efektivitu. Každý z nich se bude snažit použit svůj jazyk kdekoliv, kdykoliv a to i za cenu lehce slabšího výkonu nebo nevhodnosti vlastnosti jejich jazyka vzhledem k výsledku. {3}

2. 1 .NET

Tento Framework byl vypuštěn do světa v roce 2000 v podobě beta verze .NET 1.0 a stal se velice populární platformou pro objektově orientované programování. Rozhraní .NET Framework je nedílnou součástí systému Windows, která podporuje vytváření a spouštění aplikací a XML webových služeb nové generace. Jeho dvě hlavní složky tvoří společný jazykový modul runtime (Common Language Runtime - CLR) a knihovna .NET Framework.

Modul CLR spravuje paměť, spouštění vláken, zpracování kódu, ověření bezpečnosti kódu, kompilaci a další systémové služby. Je srdcem i duší Microsoft .NET Frameworku a obsahuje tzv. just-int-time kompilátor, kdy CLR sestavuje prostřední kód jazyka známého jako Common Intermediate Language

(CIL) do strojových instrukcí, které jsou prováděny přímo procesorem počítače. CLR také zrychluje produktivitu vývojáře. Například programátoři mohou psát aplikace v jejich zvoleném vývojovém jazyce, plně využívat modul runtime, knihovny tříd a komponent vytvořených jinými vývojáři v jiných jazycích.

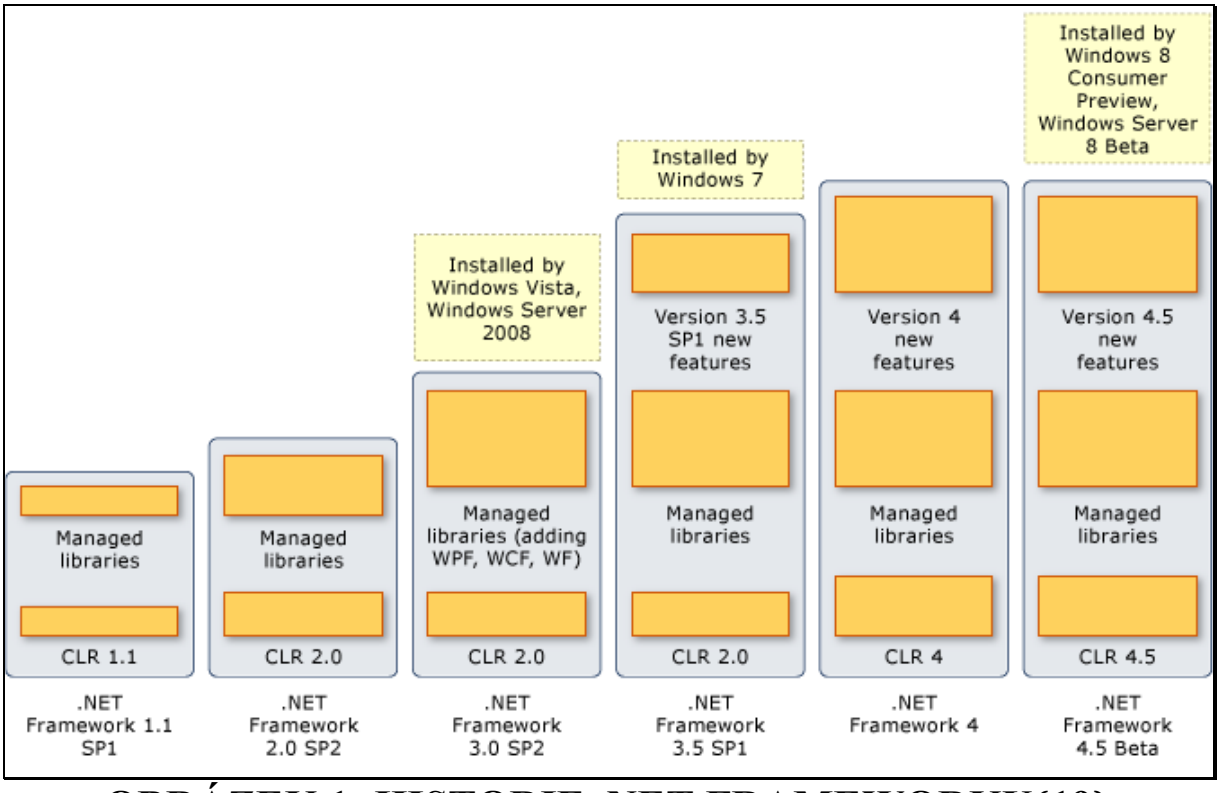
Knihovna tříd rozhraní .NET Frameworku je kolekce opakovaně použitelných typů, které jsou úzce integrovány s modulem CLR. Knihovna tříd je objektově orientovaná, poskytující typy, ze kterých může spravovaný kód odvodit funkcionalitu. Nejen toto činí rozhraní .NET Framework snadno použitelné, ale také snižuje dobu spojenou s učením nových vlastností rozhraní .NET Framework. Navíc komponenty třetích stran se mohou bezproblémově integrovat s třídami v rozhraní .NET Framework. [7]

Rozhraní .NET Framework můžete využít například k vytvoření následujících aplikací a služeb:

- Konzolová aplikace
- Aplikace s grafickým uživatelským rozhraním systému Windows
- Aplikace Windows Presentation Foundation
- Aplikace technologie ASP.NET
- Webové služby
- Služby systému Windows
- Silverlight aplikací

Aktuálně je .NET Framework dostupný ve verzi .NET Framework 4.5.

2. 1. 1 Historie .NET



OBRÁZEK 1: HISTORIE .NET FRAMEWORKU{19}

Jak je možné vidět na Obrázku 1, tak některé .NET Frameworky jsou automaticky instalovány s instalací operačního systému Windows.

.NET Framework:	CLR verze:	Visual Studio verze:	Popis:
1.0	1.0	Visual Studio .NET	Obsahoval první verzi CLR a první verzi knihovny tříd
1.1	1.1	Visual Studio .NET 2003	Zahrnoval aktualizace pro ASP.NET a ADO.NET. Tato verze byla následně dvakrát aktualizována Service Packem 1 (SP1) a SP2.
2.0	2.0	Visual Studio 2005	Představena nová verze CLR s dodatky ke knihovně tříd, včetně generiky, generických kolekcí a významných dodatků pro ASP.NET. Tato verze byla také postupně aktualizována pomocí SP1 a SP2.
3.0	2.0	Visual Studio 2005	Tato verze je v podstatě .NET Framework 2.0 s přidáním Windows Presentation Foundation (WPF), Windows Communications Foundation (WCF), Windows Workflow Foundation (WF) a CardSpace. Následně aktualizováno s SP1 a SP2.
3.5	2.0	Visual Studio 2008	Přidány nové funkce jako je AJAX a LINQ. Aktualizace pomocí SP1 přidala .NET Framework Clinet Profile, Dynamic Data a malou sadu dalších vylepšení.
4	4	Visual Studio 2010	Obsahuje novou verzi CLR, rozšířené knihovny základní třídy a nové funkce, jako je Managed Extensibility (MEF), **Dynamic Language Runtime (DLR)** a kód smluv.
4.5 Beta	4.5	Visual Studio 11 Beta	Zahrnuje vylepšenou verzi CLR, podporu pro budování Windows Metro style aplikace a vylepšení pro WPF, WCF, WF a ASP.NET.

Tabulka 1 – Historie verzí .NET Frameworku {19}

Tabulka 1 nám ukazuje, že .NET Framework si od svého vypuštění prošel již řádným vývojem. V tabulce je také možno vypozorovat, že hlavní vývojové prostředí pro .NET Visual Studio bylo svou verzí vždy spjato s konkrétní verzí .NET Frameworku. Od Visual Studia 2008 nastala změna, protože Visual Studio 2008 začalo podporovat multitargeting. Multitargeting nám umožňuje psát aplikace pro .NET Framework verzí 2.0, 3.0 i 3.5, vzhledem ke společnému jádru. Nejnovější plná verze verze Visual Studio 2010 taktéž podporuje multitargeting. Jak je již zvykem, tak s příchodem .NET 4.5 Beta, bylo vydáno také Visual Studia 11 Beta, která umožňuje vytvářet Metro style aplikace pro Windows 8.

Aktuálně je .NET Framework dostupný již v beta verzi 4.5, který umožňuje snadno využívat Windows 8 technologie, jako je Windows Runtime, přímo z .NET 4.5. Přístup k datům je jednodušší než kdy jindy s podporou nejnovějších funkcí v SQL Server a podporou pro WebSockets. Programy jsou spolehlivější, rychlejší startování ASP.NET a vylepšený server Garbage Collector. .NET 4.5 dále také vylepšuje podporu HTML5 v ASP.NET.{20}

Pro tuto práci bylo důležitým krokem zařazení DLR přímo do .NET Frameworku 4.0.

2. 2 DLR – Dynamic Language Runtime

Dynamic Language Runtime (DLR) přidává malý soubor klíčových funkcí k CLR, který jej podstatně vylepšuje. Přidá k platformě řadu služeb navržených speciálně pro potřeby dynamických jazyků, které vyžadují. Mezi tyto potřeby patří sdílený dynamický systém, standardní hostingový model a podporu pro snadné vytváření dynamického kódu. S těmito doplňky se vytváření vysoce kvalitního naimplementování dynamického jazyka v .NET stává mnohem jednodušším. Mnohem důležitějším je, aby tyto funkce u všech dynamických jazyků, které využívají DLR, bylo možné sdílet kód s ostatními jazyky tak dobře, jako se to daří již existujícím silným statickým jazykům na platformě jako je Visual Basic .NET (VB.NET) a C#. {4}

DLR také pomáhá vytvořit knihovny, které podporují dynamické operace. Například, pokud máte knihovnu, která používá XML nebo JSON objekty, mohou se objekty jevit jako dynamické objekty pro jazyky, které využívají DLR. To umožňuje uživatelům, kteří využívají tyto knihovny, napsat syntakticky jednodušší a přirozenější kód pracující s objekty a přistupující k objektovým členům.

Jako příklad je možné uvést následující kód pro inkrementaci čítače v XML:

```
Scriptobj.SetProperty("Count", ((int)GetProperty("Count")) + 1 );
```

Výpis 1: Inkrementace čítače v XML pomocí C#

Pomocí DLR můžete využít pouze následující kód, který odpovídá stejné operaci:

```
Scriptobj.Count += 1;
```

Výpis 2: Inkrementace čítače v XML pomocí DLR {21}

Tak jako CLR je i DLR částí .NET Frameworku. Instalace se provádí pomocí .NET Frameworku a Visual Studio instalačních balíčků. Open source verze DLR je také k dispozici ke stažení na webu CodePlex (www.codeplex.com/dlr).

Historie DLR se začala psát na MIX 2007 (MIX – každoročně pořádaná konference společností Microsoft pro webové vývojáře a designéry), kde Microsoft oficiálně představil DLR. Microsoft se následně rozhodl dát tento projekt jako open source a jeho vedením byl pověřen Jimmy Schementi.

DLR následně prošel 0.9 verzí vydanou 10 prosince 2008. Verze 1.0 byla vydána 16 dubna 2010. Poté Microsoft dne 16. Července 2010 změnil licenci z DLR Microsoft Public License na Apache License verze 2.0.[13] Poté jak již bylo zmíněno, tak s vydáním 4. verze .NET Frameworku bylo DLR začleněno přímo do Frameworku samotného.[7] Jak je možné vidět na obrázku 2.

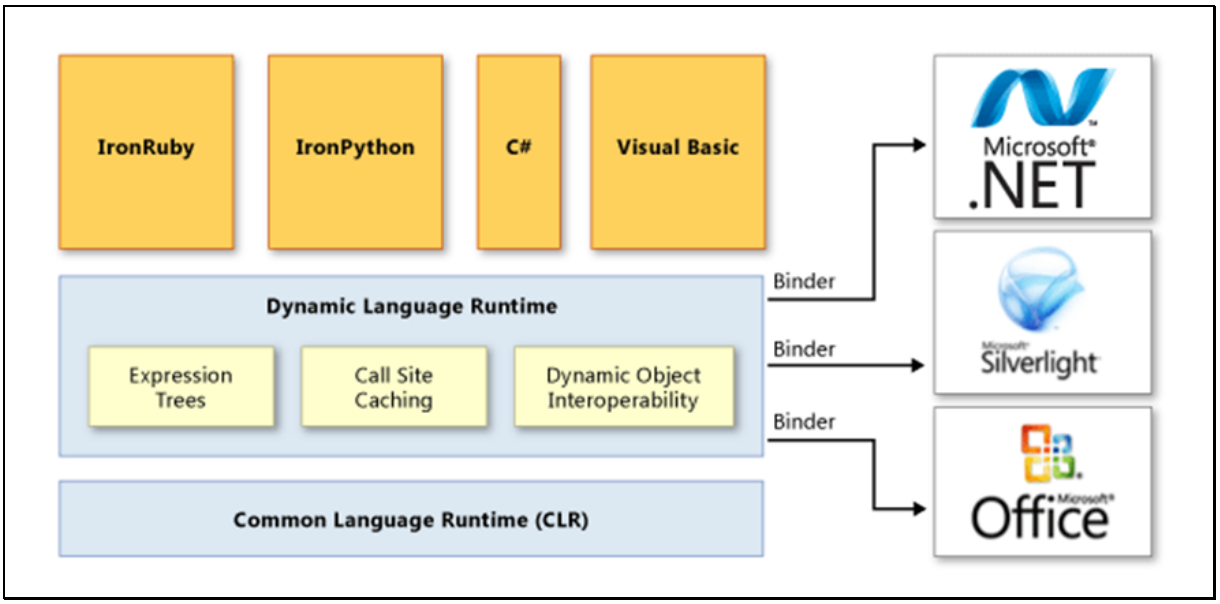
Obrázek 2: Začlenění DLR do .NET Frameworku

Hlavní výhody DLR jsou:

- Jednoduše umožnit dynamickým jazykům využívat sílu .NET Frameworku
- Umožňuje přidání dynamických vlastností v staticky typovaných jazycích
- Umožňuje sdílení knihoven a objektů

2. 3 Seznámení s architekturou DLR

Jak již bylo zmíněno DLR přidává sadu funkcí k CLR, které CLR rapidně vylepšují a hlavně umožňují využít dynamických jazyků nebo dynamických vlastností v .NET.

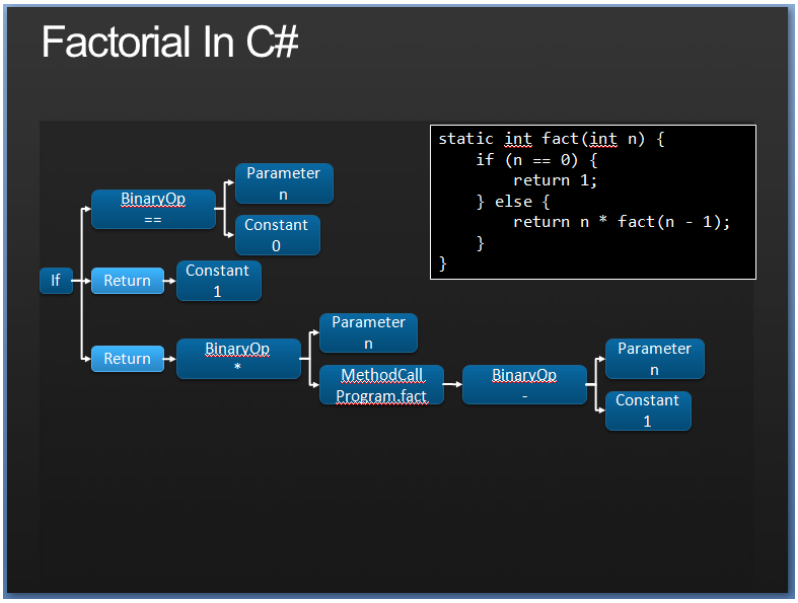


Obrázek 3: Pohled na architektonické umístění DLR{21}

Jak je možné vidět na obrázku 1, tak hlavními funkcemi DLR přidanými k CLR jsou Expression Trees, Call Site Caching a Dynamic Object Interoperability

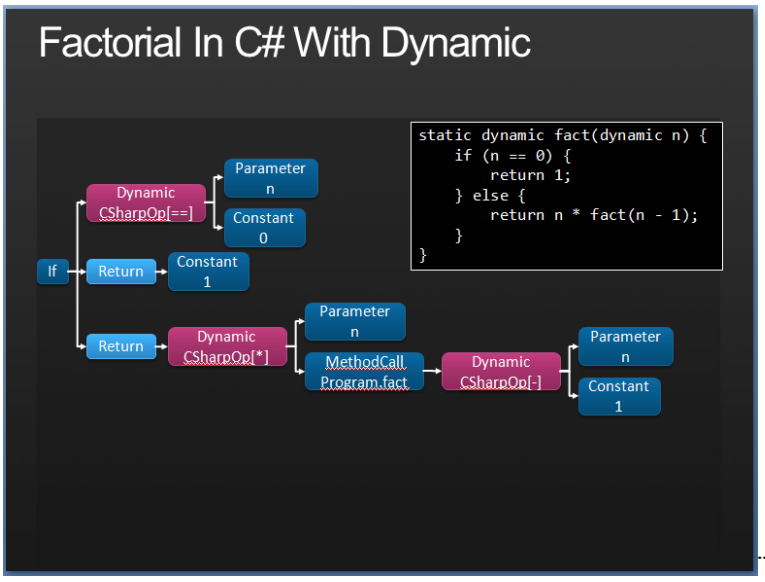
2. 3. 1 Expression Trees

Expression Trees představují kód ve stromové struktuře dat, kde každý uzel je výraz, např. volání metody nebo binární operace jako je x < y. Na obrázku 3 je možné vidět příklad metody faktoriálu v jazyce C#, zobrazenou jako Expression Tree.



Obrázek 4: Metoda faktoriál se statickými objekty v C# jako Expression Tree{22}

Pro použití DLR musí jazyk vyprodukovat expression trees, které representují jazykové úrovně kódu ve stromové struktuře a jedná se o runtime, pomocný runtime a nepovinné dynamické objekty, které implementují rozhraní IDynamicMetaObjectProviver. Na obrázku 3 jsme mohli vidět metodu faktoriál pracující se statickými objekty, jak se změní tato metoda pro pracující s dynamickými objekty je možné vidět na obrázku:



Obrázek 5: Metoda faktoriál s dynamickými objekty v C# jako Expression Tree{22}

Jak nám obrázky 3 a 4 naznačují, tak při realizaci dynamicky řešeného faktoriálu dojde ke změně pouze ve třech uzlech stromu a to pouze na místech, kde se dynamické objekty zpracovávají.

2. 3. 2 Call Site Caching

Dynamické volání stránky je místo v kódu, kde můžete provádět operace jako a + b nebo a.b() na dynamických objektech. DLR ukládá znaky a,b a informace o provozu. Pokud taková operace byla provedena dříve, DLR získá všechny potřebné informace z cache paměti pro rychlé odeslání.

2. 3. 3 Dynamic Object Interoperability

DLR poskytuje sadu tříd a rozhraní, které representují dynamické objekty, operace a mohou být využity jazykovými vývojáři a autory dynamických knihoven. Mezi tyto třídy například patří:

- **DynamicObject** – tato třída umožňuje definovat, které operace můžou být použity na dynamické objekty a jak tyto operace provést. Nelze přímo vytvořit instanci DynamicObject class, ale je z ní možné dědit.{23}

- **IDynamicMetaObjectProvider** – Jedná se o rozhraní, které implementuje třída DynamicObject a umožňuje sdílet instance DynamicObject třídy mezi podporovanými jazyky v DLR. {23}

- **DynamicMetaObject** – tato třída reprezentuje dynamickou vazbu a vazební logiku objektu účastnícího se dynamické vazby {25}

- **ExpandoObject class** – tato třída také implementuje IDynamicMetaObjectProvider a jedná se podobně o implementaci dynamického objektu, která umožňuje získání, nastavení..atd. Pokud ovšem chcete definovat typy, které mají svou vlastní dynamickou sémantiku, je nutno použít DynamicObject. {24}

3. Silverlight

Silverlight je framework pro budování tzv. Rich Browser-hosted Applications (Chytrých webových aplikací). Jednou z nejdůležitější součástí Silverlight jsou jeho bohaté možnosti pro vylepšení webových stránek, které se zatím skládají pouze z propojení HTML, CSS a JavaScriptu.

Obrázek 6: Oficiální logo Microsoft Silverlight

Pomocí technologie Silverlight můžete pečlivě a rafinovaně vytvořit a nebo přizpůsobit dynamické webové stránky s interaktivní grafikou, použít vektorové animace, přehrávat videa a zvukové záznamy. Pokud Vám tohle všechno přijde povědomé, je to tím, že o stejné se už pokoušeli technologie jako Java, ActiveX, Shockwave a nejúspěšnějším se stal Adobe Flash. Přestože se všechny tyto alternativy ještě stále využívají, žádná z nich se nestala dominantní platformou pro vývoj bohatých webových aplikací. Mnohé z nich trpí řadou problémů jako jsou otravné instalace, špatné vývojové nástroje a nedostatečnou kompatibilitou s řadou webových prohlížečů. Jediná technologie, která se dokázala těmto problémům vyhnout je Flash.
Flash se může pochlubit vynikající multiplatformní podporou a největší rozšiřitelností. Ovšem Flash se teprve nedávno vyvinul z multimediálního přehrávače do souboru dynamických programovacích nástrojů. Proto stále nabízí méně, než moderní programovací prostředí jako je .NET. [2]

Přesně zde nastupuje na scénu Silverlight. V současné době má Flash náskok, protože je nainstalován na více než 90% počítačů na světě. Nicméně Silverlight se může pochlubit několika architektonickými prvky, se kterými se Flash nemůže rovnat. Nejdůležitější je, že Silverlight je založen na zmenšené verzi CLR z .NET a díky tomu umožňuje vývojářům psát kód na straně klienta např. čistě pomocí C#. Protože Silverlight čerpá právě ze síly CLR a stává se více než jenom konkurencí pro Flash. {2}

Výhodou Silverlightu je také jeho instalace, která probíhá formou jednoduchého doplňku plug-in. Tento malý plug-in otevírá brány pro zobrazení obsahu vytvořeného z celé řady různých lidí a firem. Instalace plug-in vyžaduje pouze stažení malého balíčku a nutí uživatele pouze k jednomu bezpečnostnímu potvrzení operace.

V unixových systémech se můžeme setkat s Moonlight, který je open source implementací Silverlight , primárně pro Linux a dalších na Unix/X11 založených operačních systémech. V Září 2007 Microsoft a Novell oznámili spoluprácí, která zahrnuje přístup k testovacím sadám společnosti Microsoft pro Silverlight a distribuci mediálních balíčků pro uživatele Linuxu, které obsahují licencované mediální kodeky pro video a audio. Tento projekt se nazývá Mono {6}

3.1 Silverlight a XAML

Protože Silverlight vychází z technologie Windows Presentation Foundation (WPF) využívá pro tvorbu uživatelského rozhraní a způsobu zápisu jeho prvků stejný jazyk jako WPF, kterým je jazyk XAML.

XAML (je zkratkou pro Extensible Application Markup Language a vyslovuje se zammel) je značkovací jazyk používaný .NET objekty. Přestože XAML je technologie, která může být aplikována na mnoho různých problémových oblastech, byla původně navržena jako součást WPF, kde umožnuje Windows vývojářům tvořit bohaté uživatelské rozhraní. XAML pro svou prezentační vrstvu využívají také Silverligh aplikace.{2}

Konceptuálně XAML hraje roli hodně podobnou jako HTML a je dokonce bližší k jeho striktnějšímu bratranci XHTML. XHTML umožňuje definovat prvky, které tvoří běžnou webovou stránku. Podobně umožňuje definovat prvky XAML, který je stejně jako XHTML založený na XML, který se skládá z prvků, jenž mohou být vnořeny jakkoliv je zapotřebí.

Pro jednoduché pochopení XAML standardu slouží pár základních pravidel:{2}

- Každý element v XAML dokumentu mapuje instanci třídy Silverlight. Jméno elementu se shoduje s přesným názvem třídy. Pro příklad, element <Button> je příkaz pro Silverlight k vytvoření objektu tlačítka.

- Tak jako v každém XML dokumentu můžete vnořit jeden prvek do druhého. Nabízí také XAML, možnost každé třídě flexibilně se rozhodnout, jak vyřešit tuto situaci. Nicméně vnoření je obvyklým způsobem jak vyjádřit obal – jinými slovy, pokud se nachází Button element v Grid(mřížka) elementu, pak uživatelské rozhraní pravděpodobně bude zahrnovat Grid, který bude uvnitř sebe obsahovat tlačítko.

- Umožňuje nastavit vlastnosti každé třídy pomocí atributů. Nicméně, v některých situacích není atribut dostatečným ke zvládnutí své práce. V těchto případech je potřeba použít vnořené značky se speciální syntaxí.

Jako příklad je uvedeno porovnání vytvoření stejného ovládacího prvku, který v našem příkladu reprezentuje tlačítko:

```
Button myButton = new Button();
myButton.Background = Color.Blue;
myButton.Content = “Nápis na tlačítku”;
this.Children.Add( myButton );
```

Výpis 3: Vytvoření prvku tlačítka v jazyce C#

```
<Button Background=“Blue“>Nápis na tlačítku</Button>
```

Výpis 4: Vytvoření prvku tlačítka pomocí jazyka XAML

3.2. Historie Silverlight

V době psaní této práce je Silverlight aktuálně ve verzi 5. Kompatibilita se staršími verzemi je však stále dodržena. Historie platformy Silverlight se začala psát v roce 2007, takže se dá říci, že se jedná o novější technologii. Silverlight vychází z rysů prezenčního rozhraní Windows Presentation Foundation (WPF), které je součástí .NET Frameworku od verze 3.0.

V srpnu 2007 vydal Microsoft spolu s finální verzí Silverlight 1.0, která pro aplikační logiku využívala JavaScript také beta verzi 1.1. Tato verze byla přibližně po roce přejmenována na verzi 2.0 a bylo již možné v plné míře využívat .NET jazyky.

V červenci 2009 se objevila verze 3.0, kde jednou z nejhlavnějších novinek byla možnost běhu aplikace na klientském počítači mimo webový prohlížeč Out of Browser (OOB). Kdy se Silverlight aplikace nejprve nainstaluje do lokálního počítače, kde na rozdíl od bežných aplikací běží v izolovaném prostoru sandboxu, takže se uživatelé nemusí obávat případného škodlivého kódu. Dále se zde objevily nové ovládací prvky, například pro datové formuláře a práci s daty. Pro plynulejší zobrazení je důležitá podpora hardwarové akcelerace, která přesune část zátěže z výpočetní jednotky počítače přímo na výpočetní jednotku grafické karty. Tohle také přispívá k možnosti 3D transformace, kdy je umožněno umístit objekty a ovládací prvky libovolně do prostoru. Ke grafické stránce přibyla ještě technologie Deep Zoom, díky které je možné vykreslit obrázky s vysokým rozlišením. Se verzí Silverlight 3 přišly také novinky k podpoře videa a audia ve vysoké kvalitě. Dále je k dispozici podpora živého streamování v plném HD rozlišení v kombinaci se Smooth Streaming, který přispívá k plynulosti přehrávání obrazu bez otravného přerušení videa z důvodu nahrávání do paměti. { 5 }

V dubnu 2010 přišla Silverlight verze 4, což jen potvrzuje raketový nástup této technologie. V této verzi se k nejvýznamnějším novinkám řadí možnost využít kolečko a také pravé tlačítko myši. Nově se objevila podpora mikrofonu, webové kamery a možnost tisku. Další novinky spočívaly v novém ovládacím prvku RichTextBox, pro práci s formátovaným textem. Prvek DataGrid byl rozšířen o možnost nastavení automatické šířky sloupců a hlavně možnost kopírovat data do Excelu.{18}

3.3. Aktuální verze Silverlight a jeho budoucnost

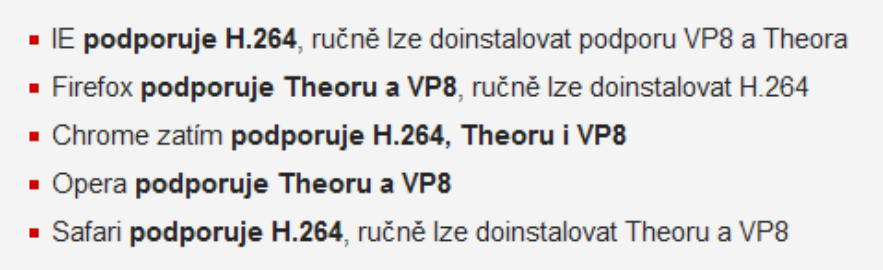
Aktuální plnou verzí je Silverlight 5 a byla vypuštěna 9 listopadu 2011 a přinesla více než 40 vylepšení.

Hlavními novinkami Silverlight 5 jsou: [10]

- Možnost umístění **breakpointu přímo v XAML** a následné ladění
- **Podpora 64 bitových systémů**
- Nové funkce pro přehrávání multimediálního obsahu
- Využití procesoru grafické karty pro realizaci 2D a 3D obsahu za pomocí hardwarové akcelerace
- Zabraňuje přehrávání chráněného obsahu v nepovolených aplikacích
- **Multicore JIT**: Rychlejší start aplikace
- **ClickCount:** Umožňuje reakci na několikanásobně stisknutí tlačítka myši
- **Variable Speed Playback** – umožňuje přehrávání videa v různých rychlostech a podporuje rychlé převinutí vpřed a vzad
- **Text Tracking & Leading**: jednoznačná kontrola nad vzdáleností mezi znaky a řádky
- **Postscript vector printing**: umožňuje vektorový tisk ()
- **Listbox/ComboBox type-ahead text searching:** Listbox a Combobox dovolují vyhledávání pomocí stisknutí písmenka

Budoucnost Silverlight značí spíše zánik, protože Microsoft oficiálně potvrdil, že HTML5 a JavaScript budou klíčovými ve vývoji pro Windows 8. Přestože Microsoft stále udržuje Silverlight, tak se na druhou stranu vehementně hlásí k HTML5 ve kterém vidí budoucnost, což dokázal s volbou pro Windows 8.

Silverlight si však prozatím jedno privilegium nechá a to přehrávání videa. HTML5 sice nabízí tag pro video, ovšem problémem je stále nejednoznačnost kodeku, na kterém se autoři prohlížečů nedokázali dodnes domluvit.



Obrázek 7: Aktuálně podporované kodeky v internetových prohlížečích

4. Dynamické jazyky v Silverlight

4. 1 Silverlight Dynamic Languages SDK

Silverlight Dynamic Language SDK tvoří most, který umožňuje spojení mezi Silverlight a DLR a díky tomuto spojení je možné psát Silverlight aplikace pomocí dynamických jazyků. Srdcem Silverlight Dynamic Languages SDK je vývojářský nástroj Chiron.exe, který vytváří Silverlight aplikační balíčky (XAP)

4. 1. 1 Historie Silverlight Dynamic Languages SDK

První verze Silverlight Dynamic Languages SDK 0.1.0 byla vydána v březnu roku 2008 a reprezentovala propojení Silverlight a DLR, kde toto spojení umožňovalo psát aplikace Silverligh jazyky IronPython, IronRuby a Managed Jscript. Tato verze odpovídala verzi Silverlight 2 Beta 1, která byla představena na MIX08 (každoročně pořádaná konference společností Microsoft pro Windows vývojáře) . Již tato verze byla vydána pod licencí Microsoft Public License, protože Microsoft předal vývoj komunitě a jeho vedením byl pověřen Jimmy Schementi.

Dále byly vydávány téhož roku verze Silverlight Dynamic Languages SDK 0.2.0 v červnu a verze 0.3.0, které kromě pár vylepšeních nenabízely závratnějších novinek a korespondovaly s verzí Silverlight 2 Beta 2. Roku 2008 stihla být vydána ještě také verze 0.4.0, již korespondovala s plnou verzí Silverlight 2.

Pak se tento rapidní vývoj trošku zpomalil, aby mohla v březnu roku 2009 přijít verze Silverlight Dynamic Languages SDK 0.5.0. Tato verze korespondovala stále se Silverlight 2, ale již také s verzí Silverlight 3 Beta, která byla představena na MIX09. Duležitým zlomovým momentem bylo stažení Managed JScriptu z tohoto balíčku a o jeho dalším vývoji se neuvažovalo.

Tento moment taky ovlivnil další vývoj samotného Silverlight Dynamic Languages SDK, protože právě díky odstranění Managed JScriptu se vývoj Silverlight Dynamic Languages SDK rozdělil na dva samostatné tábory, které se dále věnovaly pouze každý svým vlastním jazykům. Vznikly dva projekty, které sice využívají Silverlight Dynamic Languages SDK, ale jsou vyvíjeny paralelně.

Dalším významným bodem, který nastal již v době paralelního vývoje IronPython a IronRuby, byl v roce 2010 odchod programového ředitele, kterým byl Jimmy Schementi a projekt na CodePlexu prakticky dlouho umíral. IronPython byl spasen a to i díky větší popularitě Pythonu oproti Ruby, když vedení projektu IronPython převzal Jeff Hardy. Iron Python tedy dále vzkvétá a poslední aktualizace byla vydána v březnu roku 2012. Oproti tomu projekt IronRuby se dostal do problémů a v současnosti stojí hlavně na komunitě a na práci právě bývalého programového ředitele Jimmyho, který i přes své vlastní působení jinde občas přispěje a obstará aktualizace IronRuby. V současné době je posledním vydáním IronRuby verze 1.1.3, která byla vydána právě za pomocí Jimmyho v březnu 2011.

4. 1. 2 Chiron

Silverlight aplikace jsou distribuovány v balíčcích respektive v XAP souborech, jenž jsou soubory typu ZIP doplněny několika potřebnými soubory. Jedná se o požadavky aplikací, napsaných pomocí dynamických jazyků, aby DLR a jazykové sestavy byly uvnitř balíčku, spolu se všemi ostatními skripty, které aplikace vyžadují. [9]

Chiron umí vytvářet XAP soubory pro dynamické aplikace, ale také hlavně pracuje jako server, který umožňuje využít své aplikace ze souborového systému při jejich vývoji. Protože Chiron běží pod Mono, je jej možné použít také na strojích s operačním systémem Mac.{3}

Základním příkazem pro příkazový řádek k vytvoření XAP souboru pomocí Chironu je:

```
bin\Chiron /d:nazev_slozky /z:vase_aplikace.xap
```

Výpis 5: Vytvoření XAP souboru pomocí Chironu ve Windows

```
mono bin/Chiron.exe /d:nazev_slozky /z:vase_aplikace.xap
```

Výpis 6: Vytvoření XAP souboru pomocí Chironu v MAC OS

Jak již bylo zmíněno, tak mnohem důležitějším využitím nástroje Chiron je pro testování aplikace ze souborového systému, bez nutnosti vytvářet další XAP soubor. Pokud se aplikace jmenuje app xapp, potom by se její aplikační soubory měly nacházet v adresáři pojmenovaném app. Chiron bude fungovat jako lokální server a dynamicky vytvářet XAP soubory tak, jak budou zapotřebí.

```
Chiron /w
```

Výpis 7: Příkaz pro spuštění Chironu

Dle výchozího nastavení bude aplikace dostupná na localhostu s portem 2060. Nejdůležitější částí XAP souboru je vstupní bod, který v dynamických aplikacích bude soubor s názvem app.py nebo app.rb, kdy bude záležet na použitém dynamickém jazyce.{3}

4. 2 Vývojové nástroje

Protože se tato práce týká technologie Silverlight aplikací, je důležité popsat si nástroje pro jejich tvorbu. Prvním je vývojové prostředí Microsoft Expression Blend, které je v současné době dostupné ve verzi 4. Druhým prostředím je Visual Studio, aktuálně Visual Studio 2010.

Aplikace napsané v jednom z těchto prostředí se může bez problému kdykoliv otevřít v druhém vývojovém prostředí a dále upravovat. Díky této vlastnosti je možné psát např. vizuální vrstvu v jednom prostředí a pro aplikační kód využít prostředí druhé a to přestože oba tyto nástroje umožňují rozdělit si vývoj Silverlight aplikace na návrh jejího vizuálního vzhledu a kódu aplikační logiky. Z vizuálního pohledu je pro návrh grafického návrhu, animací a 3D grafiky komfortnější vytvořit tuto část pomocí prostředí Expression Blend.{2}

Protože práce pojednává o využití dynamických programovacích jazyků, které se využívají pro psaní aplikační logiky, je pro tuto práci podstatnější vývojové prostředí Visual Studio 2010. Microsoft Expression Blend je možno a bude v tomto případě využito pouze k tvorbě prezentační vrstvy.

4. 2. 1 Microsoft Visual Studio

Micorosoft Visual Studio 2010 poskytuje nástroje pro úplné začátečníky až po profesionály a velice zkušené vývojáře. Samotné Visual Studio 2010 v základní instalaci umožňuje vytvářet pouze Silverlight aplikace ve verzi 3.0. Pro vývoj aplikací Silverlight 4.0 je zapotřebí stáhnout rozšiřující balík Silverlight 4 Tools for Visual Studio 2010. Pro vytváření aplikací v nejnovější verzi Silverlight 5

je zapotřebí, podobně jako u Silverlight 4, stáhnout balík Silvelight 5 Tools for Visual Studio 2010 SP1 (Tento balík je určen již pro Visual Studio s nainstalovaným Service Packem 1).

4. 2. 1 Microsoft Expression Blend

Návrhové prostředí Micorosft Expression Blend je flexibilní a produktivní grafické vývojové prostředí. Pomáhá při tvorbě moderních a vizuálně propracovaných aplikací s interaktivní podporou 3D zobrazování a přehrávání multimédií. Umožňuje vytvoření a úpravy prezentační vrstvy graficky bohatých aplikací, ať už webových, využívajících technologii Silverligt, nebo kladských desktopových aplikací založených na technologii Windows Presentation Foundation (WPF). {18}

4. 3. IronPython v Silverlight

4. 3. 1 Python

Jedná se o dynamický objektově orientovaný skriptovací programovací jazyk, který v roce 1991 navrhnul Guido van Rossum. Python je vyvíjen jako open source projekt, který zdarma nabízí instalační balíky pro většinu běžných platforem (Unix, Windows, Mac OS) a ve většině distribucí systému Linux je Python součástí základní instalace. [14]

K význačným vlastnostem jazyka Python patří jeho jednoduchost z hlediska učení. Bývá dokonce považován za jeden z nejvhodnějších programovacích jazyků pro začátečníky. Tato skutečnost je dána tím, že jedním z jeho silných inspiračních zdrojů byl programovací jazyk ABC, který byl jako jazyk pro výuku a pro použití začátečníky přímo vytvořen. Python ale současně bourá zažitou představu, že jazyk vhodný pro výuku není vhodný pro praxi a naopak. Podstatnou měrou k tomu přispívá čistota a jednoduchost syntaxe, na kterou se při vývoji jazyka hodně dbá. [8]

Další význačnou vlastností jazyka Python je produktivnost z hlediska rychlosti psaní programů. Týká se to jak nejjednodušších programů, tak aplikací velmi rozsáhlých. U jednoduchých programů se tato vlastnost projevuje především stručností zápisu. U velkých aplikací je produktivnost podpořena rysy, které se používají při programování ve velkém, jako jsou například přirozená podpora jmenných prostorů, používaní výjimek, standardně dodávané prostředky pro psaní textů a dalšími. S vysokou produktivností souvisí dostupnost a snadná použitelnost široké škály knihovních modulů, umožňujících snadné řešení úloh z řady oblastí. [14]

Jako základní příklad jazyka Python je uveden výpis hlášky Hello World na standardní výstup:

```
print “Hello World”
```

Výpis 8: Příkaz pro vypsání hlášky Hello World

Pro porovnání můžeme použít stejný příklad v programovacím jazyce C#:

```
public class Hello
{
    public static void Main(string[ ] args)
    {
        System.Console.WriteLine("Hello World!");
    }
}
```

Výpis 9: Příkaz pro vypsání hlášky Hello World

Rozdíl mezi oběma aplikacemi je na první pohled patrný. Python dynamicky interpretovaný jazyk, kde interpret jen vyhodnotí jednotlivé řádky kódu a prezentuje výsledek. Proto se jedná o zjednodušený a velice přehledný kód.

Protože v dnešní době se nejvíce využívá objektového programovaní, které oba jazyky podporují, proto se nabízí příklad objektové programování kódu pro Hello World. Viz. Výpis 10:

```
class Hello( object ):
    def __init__ ( self, msg = 'Hello World' ):
        self.msg = msg

    def SayHello ( self ):
        print self.msg

app = Hello()
app.SayHello()
```

Výpis 10: Objektově řešený příklad Hello World pomocí Python

```
using System;
class Hello
{
    private string _msg;
    public Hello()
    {
        _msg = "Hello World";
    }
    public Hello( string msg )
    {
        _msg = msg;
    }
    public void SayHello()
    {
        Console.WriteLine( _msg);
    }
    public static void Main()
    {
            Hello app = new Hello();
            app.SayHello();
    }
}
```

Výpis 11: Objektově řešený příklad Hello World pomocí C#

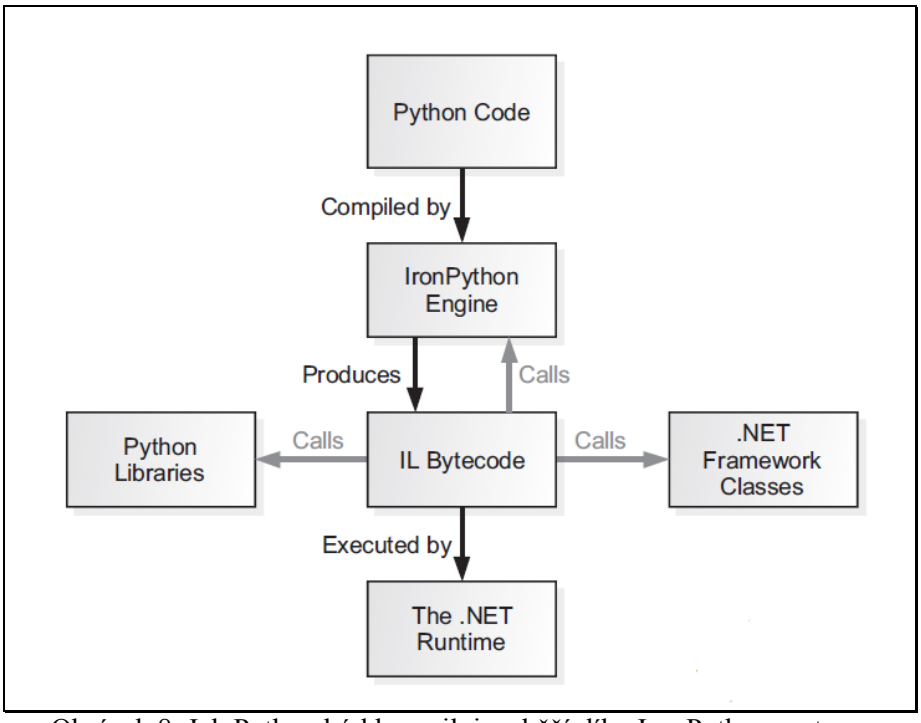
Uvedené příklady objektového řešení příkladu Hello World ještě názorně prezentují hlavní výhodu Pythonu, kterou je dobrá čitelnost a striktní jednoduchost zdrojového kódu.

4. 3. 2 IronPython

IronPython je napsaný v C# a jedná se o open source implementaci jazyka Python pro .NET. Byl vyvinut společností Microsoft jako součást plánu jak udělat z CLR lepší platformu pro dynamické jazyky. IronPython 1 odpovídá verzi Python 2.4 a IronPython 2 verzi Python 2.5. IronPython je Python.

Aktuální verzí je IronPython 2.7.2.1. Tato verze v sobě obsahuje všechno z verzí IronPython 2.7 a 2.7.1. Nejnovějšími vylepšení jsou podpora pro ZIP archívy pomocí zipimort module, dále nabízí například sqlite3 module a mnoho dalších.

IronPython se primárně skládá z IronPython motoru a dalších nástrojů. IronPython motor sestavuje Python kód do IL, který běží na CLR. IronPython může být zkompilován do assembly, které lze uložit na disk a použít k tvorbě pouze binárních distribucí aplikací.



Obrázek 8. Jak Python kód kompiluje a běží díky IronPython motoru.

Vývojové cykly jsou obvykle rychlé s využitím Pythonu. S dynamicky typovanými jazyky lze dosáhnout úkolu s kratším kódem, což dělá IronPython ideálním pro prototypy aplikací nebo skriptování administrátorských úkolů, které Vás nemohou stát hodně času.

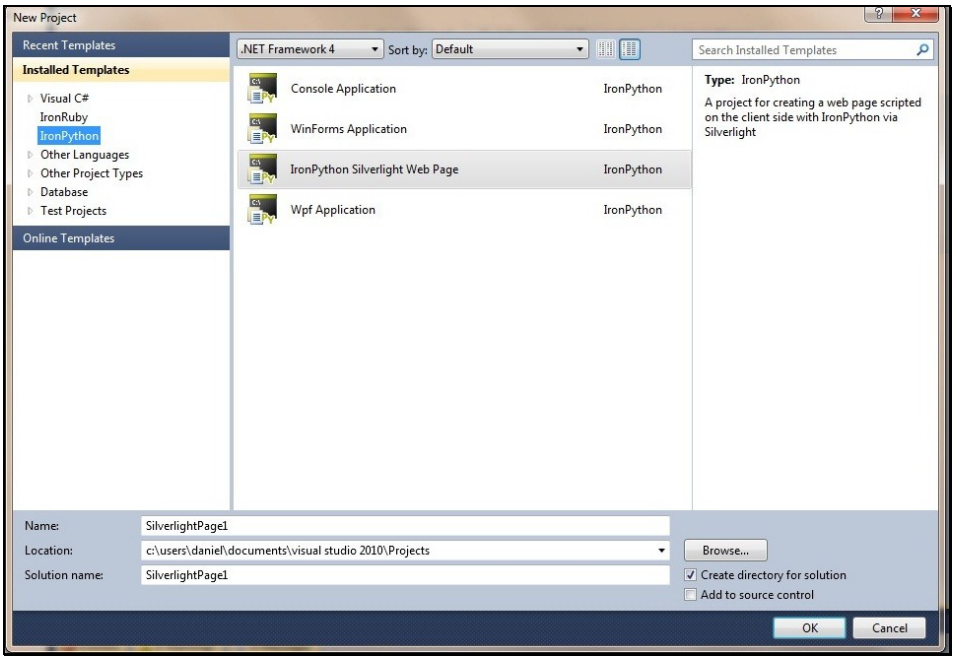
4. 3. 3 Instalace IronPython

Vývojáři .NET se nemusí bát, že by při vývoji aplikací pomocí IronPython museli pracovat s neznámým vývojovým prostředím. Nejjednodušším způsobem pro instalaci a následné vývoj aplikací pomocí IronPythonu je stažení kompletního instalačního balíčku. Tento balíček, který je v současnosti dostupný ve verzi 2.7.2.1, obsahuje kompletní instalaci IronPythonu včetně nástrojů pro Visual Studio 2010. Balíček je dostupný na stránkách projektu:
(http://ironpython.codeplex.com/releases/view/74478)

Obrázek 9: Instalace IronPython

Ovšem je třeba si dát pozor na současnou nekompatibilitu při instalaci SP1 pro Visual Studia 2010 (VS2010), který také vyžaduje balíček pro kompatibilitu a vývoj Silverlight verze 5. IronPython instalátor je zkonstruován tak, aby nainstaloval IronPython, včetně integrace do VS2010, ale instalce SP1 změní některé vlastnosti VS2010, se kterými však instalátor IronPythonu nepočíta a apliace nebude možné spouštět. Jednou z konkrétních chyb je instalace Chironu, který následně není schopný zprovoznit server pro IronPython aplikaci. Řešením tohoto problému je instalace IronPythonu v době nainstalovaného VS2010 bez SP1 a jeho instalaci uskutečnit s již nainstalovaným IronPythonem.

Po instalaci můžete Visual Studio využít k vytvoření IronPython projektů s plným přístupem k designéru a debuggru, ale hlavně budete mít možnost tvorby Silverlight aplikací pomocí IronPython.



Obrázek 10: Nový IronPython Silverlight projekt ve VS2010

4. 4 IronRuby v Silverligh

4. 4. 1 Ruby

Ruby je jazykem pečlivé rovnováhy. Vytvořil jej Yukihiro Matsumoto, který smísil části jeho oblíbených jazyků (Perl, Smalltalk, Eiffel, Ada a Lisp) a vytvořil nový jazyk. [6]

Od svého veřejného vydání roku 1995 se Ruby stával oblíbeným u mnoha vývojářů po celém světě, získal si solidní základnu a velikou komunitu, a proto v roce 2006 nastalo masové přijetí Ruby ve světě. V době tohoto vývoje Matsumoto stále hledal další výhody v jiných jazycích a v rozhovoru, který poskytl 29. listopadu 2001 prohlásil: „Chtěl bych skriptovací jazyk, který bude silnějším než Perl a více objektově orientovaným než je Python.“

V Ruby je všechno objektem. Každému kousku informace a kódu můžou být přiřazeny jejich vlastnosti a akce. Objektově orientované programování volá vlastnosti instance reprezentované jejím názvem a akce jejími metodami. To, že je Ruby natolik objektově orientovaným jazykem, lze nejlépe vidět na práci s čísly. V mnoha ostatních programovacích jazycích totiž čísla a další primitivní typy nejsou objekty. Ruby se v tomto nejvíce inspirovalo jazykem Smalltalk, který přidává metody a proměnné instance pro všechny své typy. [6]

Jako příklad pro tvrzení, že v Ruby je všechno projekt slouží přiklad:

```
1 + 2
```

Výpis 12: Příklad pro sčítaní dvou čísel pomocí Ruby

Tento případ je vlastně volání metody pro sčítaní nad objektem s jedním argumentem:

```
1.+(2)
```

Výpis 13: Ekvivalentní kód pro 1+2 viz.Výpis12:

Oba objekty 1 i 2 jsou objekty třídy Fixinum, která v Ruby slouží k uchování celých čísel. Pro větší čísla slouží třída Bignum, jejíž teoretickou hranicí je velikost operační paměti.

Ruby je novějším jazykem než Python a dlouho jeho rozlet komplikovala absence kvalitní dokumentace v anglickém jazyce. Dnes už je však anglických materiálů celkem dost a zájem o Ruby narostl také díky úspěchů webového frameworku Ruby on Rails. V únoru aktuálního roku byla vydána zatím poslední verze Ruby 1.9.3-p125.

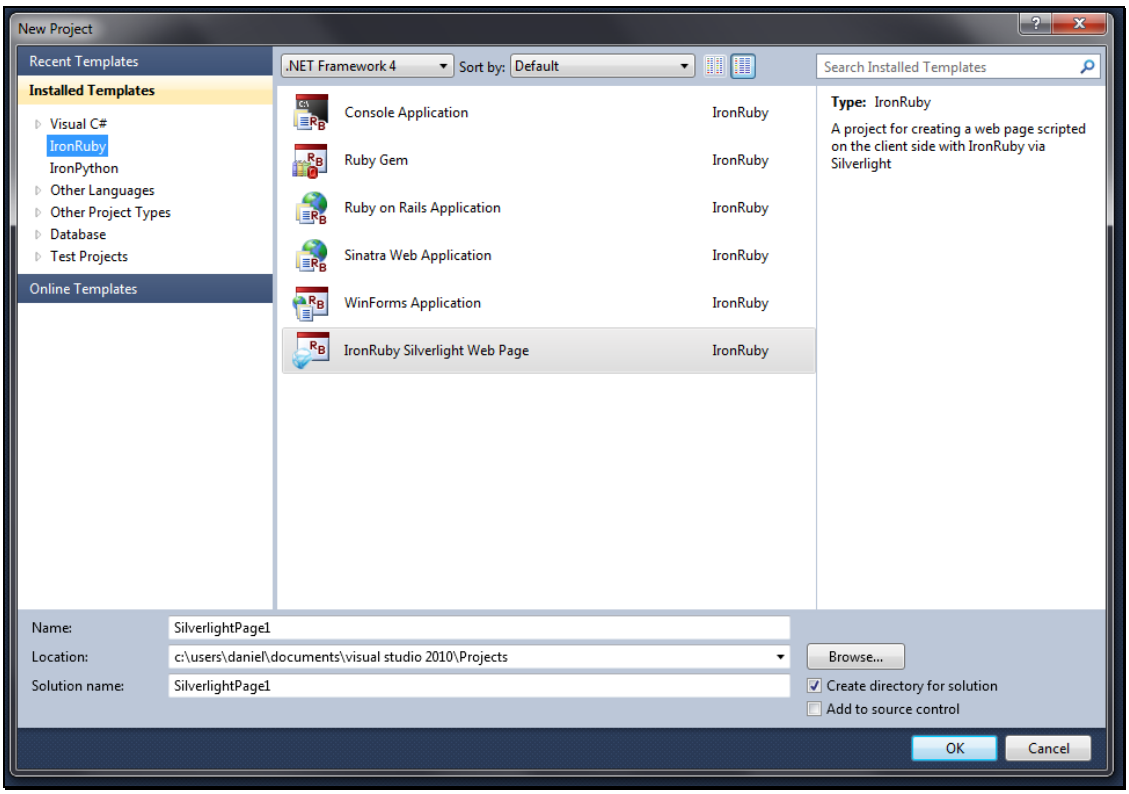
4. 4. 2 IronRuby

Jedná se podobně jako IronPython o open source implementaci, ale tentokrát programovacího jazyka Ruby, která umožňuje psát .NET aplikace pomocí Ruby.

Aktuální verzí je IronRuby 1.1.3, která vylepšuje kompatibilitu s verzí Ruby 1.9.2, která byla vydána v březnu roku 2011. Tato verze v sobě obsahuje také balíček pro integraci do vývojového prostředí Visual Studio 2010. V této verzi se komunita vývojářů rozhodla upustit od kompatibility s verzí Ruby 1.8.6, kterou naplno obsahovala verze IronRuby 1.0. Komunita tedy doporučuje využít IronRuby 1.0 pro kompatibilitu Ruby 1.8.6.

4. 4. 3 Instalace IronRuby

Také instalace dynamického jazyka IronRuby pro .NET a Silverligh se provádí pomocí kompletního instalačního balíčku, který je k dispozici stránkách projektu Codeplex (http://ironruby.codeplex.com/releases/view/60511). A výsledkem této instalace je možnost vyvíjet Silverlight aplikace ve Visual Studiu 2010.



Obrázek 11: Nový IronRuby Silverlight projekt ve VS2010

Jak je možné vidět na obrázku 12, tak instalace IronRuby pro VS2010 následně nabízí možnost psát aplikace také ve frameworku Ruby on Rails, který, jak již bylo zmíněn, nejvíce proslavil Ruby.

Také při instalaci IronRuby je potřeba brát v úvahu problém, který již byl zmíněn. Tento problém spočíval v instalaci IronPythonu v době nainstalovaného SP1 pro VS2010. Proto je potřeba instalovat také IronRuby již před instalací SP1 pro VS2010.

5. Silverlight aplikace pomocí dynamických jazyků

Tato kapitola přiblíží první aplikace v Silverlight pomocí dynamických jazyků IronPython a IronRuby. Aplikace Ukázková kalkulačka bude obsahovat také implementaci v jazyce C#, pro možnost zhodnocení implementace v jednotlivých jazycích.

5. 1 První aplikace

Pro úplně první aplikace s IronPython a IronRuby nám postačí textový editor a webový prohlížeč.

5. 1. 1 Hello from IronPython

Umožnit HTML stránce, aby spouštěla skripty za pomocí jazyka IronPython, je velice jednoduché.

První možností je přidání reference na soubor DLR v JavaScriptu do hlavičky HTML:

```
<script src=”http://getstalt.ironpython.net/dlr-latest.js”
type=”text/javascript”></script>
```

Výpis 14: Přidání refernce na aktuální DLR JavaScript soubor

Druhá možnost je spjata s požadavkem vyvíjet a spouštět Silverlight aplikace bez nutnosti internetového připojení:

```
<script type=”text/javascript”>
    Window.DLR = {path: ‘path/to/gestalt.latest’}
</script>
<script src=”path/to/getstalt.latest/dlr.js” type=”text/javascript”>
</script>
```

Výpis 15: Přidání reference na DLR soubor, umístěným dle specifikované cesty

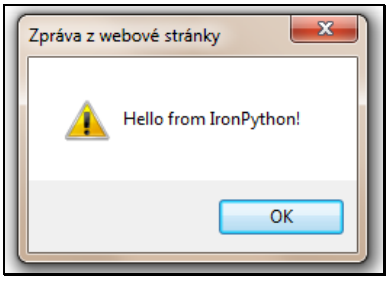
V závislosti na verzi getstalt.ironpython.net je doporučeno dlr-latest.js, který nabízí jistotu nejnovější verze IronPythonu. Můžete si ovšem zvolit specifickou verzi, pro zajištění stability např. dlr-20100305.js.

Po přidání reference již bude možné vkládat a spouštět Python skripty v HTML:

```
<script type= “type/python”>
    Window.Alert(“Hello from IronPython!”)
</script>
```

Výpis 16: Zobrazení hlášky Hello from IronPython!

Výsledek aplikace Hello from IronPython je možné vidět na obrázku 13.



Obrázek 12: Výsledek aplikace Hello from IronPython

5. 1. 2 Hello from IronRuby

Umožnění spouštění Ruby skriptů je obdobné jako u IronPython.

Přidání reference na aktuální DLR JavaScriptový soubor:

```
<script src=”http://getstalt.ironruby.net/dlr-latest.js” type=”text/javascript”></script>
```

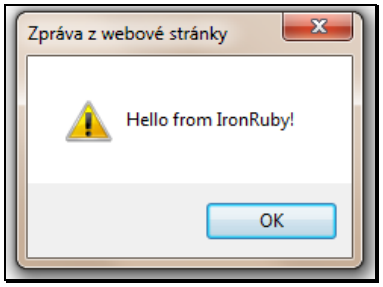
Výpis 17: Přidání reference na aktuální DLR JavaScript soubor

Umožněno je také přidání reference na DLR JavaScriptový soubor na lokálním disku, který je totožný jako u IronPython (Výpis:). Pouze specifikovaná cesta vede k DLR JavaScript souboru pro IronRuby.

Po přidání reference již bude možné vkládat a spouštět Ruby skripty v HTML:

```
<script type= “type/ruby”>
        Window.Alert(“Hello from IronRuby!”)
</script>
```

Výpis 18: Zobrazení hlášky Hello from IronRuby!



Obrázek 13: Výsledek aplikace Hello from IronPython

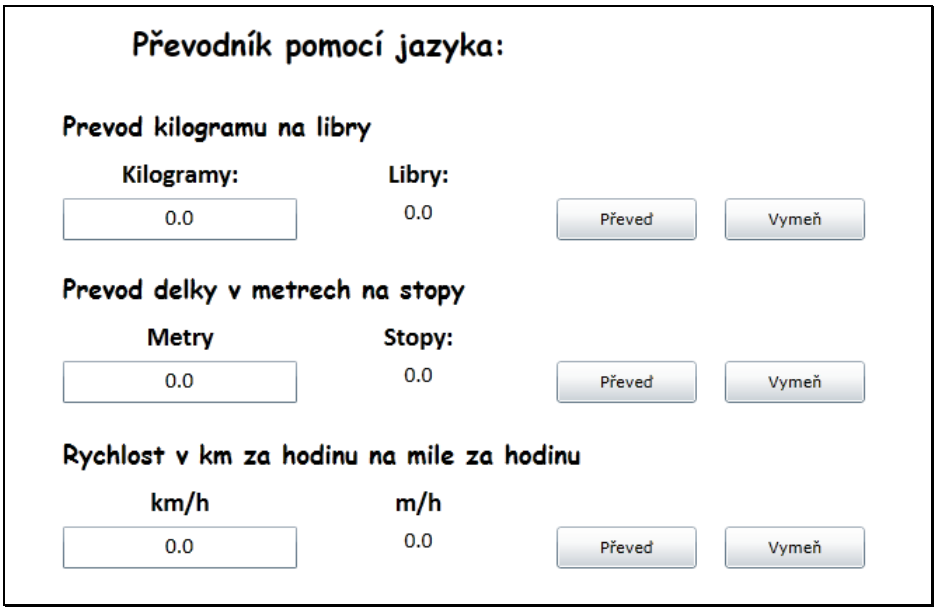
5. 2 Aplikace převodníku

V tomto případě již bude prezentována síla spojení jazyka XAML a Silverlight. Každá Silverlight aplikace Převodník nezávisle na programovacím jazyce, kterým byla vytvořena, bude pracovat s grafickým návrhem, reprezentovaným XAML souborem bokem.xaml. Tento soubor byl vytvořen v nástroji Microsoft Expression Blend 4, bude pro všechny jazyky prezentační vrstvou a budou s ním pracovat. Tímto bude poukázáno nejenom na jednoduchost a přehlednost grafického návrhu pomocí XAML, ale také jeho nezávislost na programovacím jazyce.

Tento příklad udává základní náhled na implementaci dynamických jazyků v aplikacích Silverlight. Bude objasněno spojení aplikačních skriptů přímo s XAML šablonou.

Úkolem aplikace Převodník je simulovat převody základních důležitých jednotek. V tomto případě půjde o převod hmotnosti mezi kilogramy a librami, pro délku je možnost převodu mezi metry a stopami a pro rychlost je uveden převod mezi kilometrem za hodinu a mílí za hodinu.

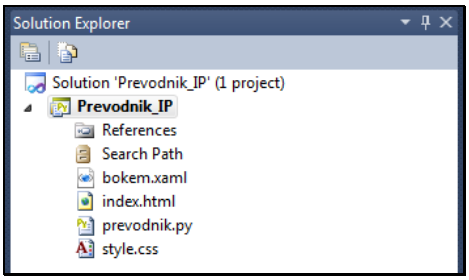
Výsledný vzhled šablony, kterou je soubor app.xaml je možné vidět na obrázku:



Obrázek 14: Šablona prezentační vrstvy aplikace Převodník

5. 2. 1 Převodník IronPython

Aplikace Převodníku v IronPython se bude skládat ze souboru index.html, jenž reprezentuje webovou stránku, která obsahuje šablonový soubor prezentační vrstvy bokem.xaml a bude skriptována pomocí Python souboru převodník.py.



Obrázek 15: Seznam souborů pro aplikaci Převodník v IronPython

Prvním krokem je potřeba přidat do index.html referenci na DLR soubor (viz. Kapitola 5.1.1). Druhým krokem je nutnost přidání reference skriptovacího souboru převodník.py na šablonový soubor bokem.xaml:

Druhým nejdůležitějším krokem je přidání reference pro Python skript a XAML soubor:

```
<script type="application/xml+xaml" src="bokem.xaml" id="prevodnik"></script>

<script type="text/python" class="prevodnik" src="prevodnik.py">
```

Výpis 19: Přidání reference pro Python skript k souboru XAML

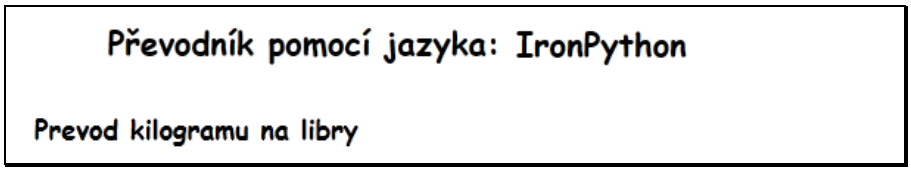
Přidání reference je v tomto případě reprezentováno klíčovými atributy class a id. Atribut class nastavíme pro skript aplikační logiky a atribut id pro prezentační vrstvu. Oba atributy pro splnění reference se musí jmenovat stejně. Po přidání reference bude již možné v souboru aplikační logiky přistupovat k objektům šablony

Jako příklad si pomocí souboru prevodnik.py nastavíme atribut Text objektu s názve jazyk:

```
xaml.jazyk.Text = 'IronPython'
```

Výpis 20: Nastavení atributu Text objektu s názvem "jazyk" na text IronPython

Výsledkem bude změnění hodnoty Text u objektu s názvem jazyk.



Obrázek 16: Výsledná změna objektu jazyk

Aplikace dále dle zadaných hodnot uživatele vypočítává převod jednotek, kde výpočet je realizován stiskem tlačítka Převeď. Příkladem obsluhující metody tlačítka pro převod je:

```
def vysl_vaha(s,e):
    global vaha

    try:
        pom = float(xaml.in_vaha.Text)
        if(vaha == 0):
            xaml.out_vaha.Text = '%f' % ( pom * 2.205 )
        else:
            xaml.out_vaha.Text = '%f' % (pom / 2.205)
    except ValueError:
        xaml.out_vaha.Text = 'error'
```

Výpis 21: Obslužná metoda pro tlačítko Převeď

Tato metoda kontroluje globální proměnnou vaha, která reprezentuje příznak pro převod kg na libry nebo opačně. Následně je hodnota vstupního parametru převedena z řetězce na datový typ float, který reprezentuje číslo s plovoucí řádovou čárkou. Výsledné číslo je převedeno na řetězec a zobrazeno pomocí objektu out_vaha. Metoda je obalena do bloku try, který hlídá ošetření výjimky ValuError, která nastává při špatně zadaném formátu čísla a reakcí na tuto výjimku je nastavení objektu out_vaha atribut Text na hodnotu error. Viz. Obrázek:

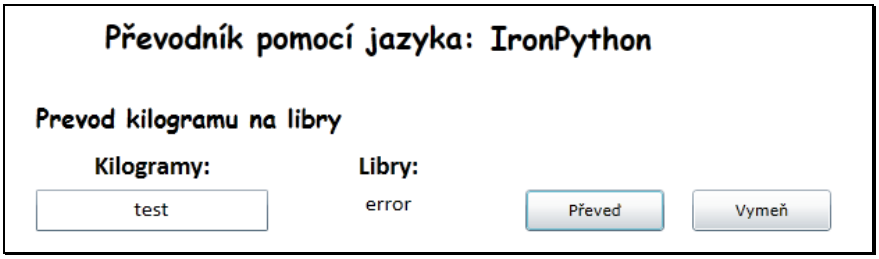
Posledním krokem k realizaci obsluhy tlačítka je zaregistrování události po jeho stisku:

```
xaml.prev_vaha.Click += vysl_vaha
```

Výpis 22: Zaregistrování události Click pro objekt tlačítka s názvem prev_vaha

Poté při každém stisknutí tlačítka bude zavolána obslužná metoda vysl_vaha. Výsledný převod 88kg na libry v Převodník IronPython je možné vidět na obrázku:

Obrázek 17: Výsledný převod 88kg na libry



Obrázek 18: Výsledný převod pro zadanou hodnotu test

5. 2. 2 Převodník IronRuby

Aplikace Převodník v IronRuby se podobně jako Převodník IronPython skládá ze souboru index.html, jenž reprezentuje webovou stránku, která obsahuje šablonový soubor prezentační vrstvy bokem.xaml a bude skriptována pomocí Ruby souboru převodník.rb

Prvním krokem je přidání reference na DLR viz. Kapitola 5.1.2. Dále je nutné přidání reference skriptovacího souboru na šablonový soubor obdobně jako u IronPython:

```
<script type="application/xml+xaml" src="bokem.xaml" id="prevodnik"></script>

<script type="text/ruby" class="prevodnik" src="prevodnik.rb">
```

Výpis 23: Přidání reference pro Ruby skript k souboru XAML

Nyní je možné stejně jako v 5.2.1 přistupovat k objektům v Ruby skriptu pomocí objektu xaml. Převodník IronRuby si tedy jako IronPython viz.Výpis: nastaví atribut Text objektu jazyk, tentokrát ovšem se svým jménem.

Pro obsluhu převodu tlačítka Převeď slouží v tomto případě metoda:

```
xaml.prev_vaha.Click do |sender, args|

    pom = (xaml.in_vaha.Text).to_f()

    if($vaha == 0)
       xaml.out_vaha.Text = ( pom * 2.205 ).to_s()
    else
       xaml.out_vaha.Text = (pom / 2.205).to_s()
    end

end
```

Výpis 24: Obsluha tlačítka Převeď v IronRuby

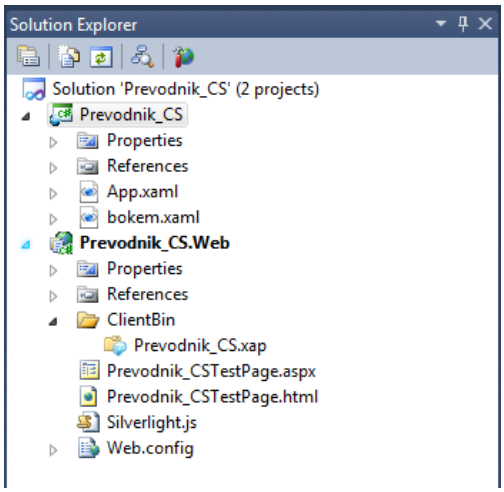
Tato metoda oproti IronPython provádí zároveň zaregistrování a zároveň obsluhu stisknutého tlačítka. Obslužná metoda oproti aplikaci Převodník IronRuby nevyžaduje nutně ošetření výjimky ValuError, protože metoda .to_f() uloží do pomocné proměnné pom číslo, které dokáže převést, než nastane chyba, bez vyvolání výjimky. Například po zadání čísla pro převod v hodnotě 100chyba, se uloží do proměnné pouze hodnota 100.

5. 2. 3 Převodník C#

Visual Studio 2010 při vytváření nového projektu nabízí několik šablon pro Silverlight aplikace:

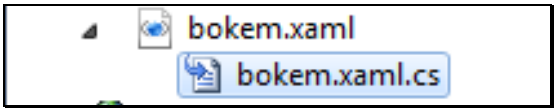
- Silverlight Application,
- Silverlight Navigation Application,
- Silverlight Class Library,
- Silverlight Business Application,
- WCF RIA Services Library,
- Silverlight Unit Test Application.

Po výběru typu aplikace následuje otázka ohledně způsobu spouštění Silverlight aplikace. Protože se jedná o webovou aplikaci, je jednou z možností přidání hostující stránky ASP.NET, které bylo využito při vytvoření aplikace Převodník C#.



Obrázek 19: Seznam souborů pro aplikaci Převodník C#

Další změnou v C# Silverligt aplikaci je využití tzv. Partial Classes, které nám umožnují rozdělit definici tříd do několika částí, které jsou spojeny při kompilaci aplikace.



Obrázek 20: Aplikační logika Převodník C#

Jak je možné vidět na obrázku 21, bude k prezenční vrstvě bokem.xaml připojena aplikační logika realizována bokem.xaml.cs. Odpadá tedy nutnost referencí pro objekty z prezenční vrstvy do aplikační logiky.

Partial Class v tomto případě bude vypadat takto:

```
namespace Prevodnik_CS
{
   public partial class Bokem : UserControl
    {
```

Výpis 25: Partial Class aplikace Převodník C#

Obsluha tlačítka pro převod

```
private void prev_vaha_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                double pom = Convert.ToDouble(in_vaha.Text);

                if (vaha == 0)
                {

                    out_vaha.Text = Convert.ToString(pom * 2.205);
                }
                else
                {
                    out_vaha.Text = Convert.ToString(pom / 2.205);
                }
            }
            catch (FormatException er)
            {
                out_vaha.Text = "error";
            }
        }
```

Výpis 26: Obslužná metoda pro stisk tlačítka prv_vaha

Obslužná metoda stejně jako aplikace Převodník IronPython musí řešit vyvolání výjimky při špatném zadání formátu čísla. Proto i zde je kód obalen blokem try a reakci na výjimku obsluhuje blok catch, který obdobně jako u Převodník IronPython nastaví objektu out_vaha atribut Text na hodnotu error.

Zaregistrování metody pro obsluhu stisknutí tlačítka je realizováno přímo XAML definici tlačítka:

```
    <Button x:Name="prev_vaha" Content="Převeď" HorizontalAlignment="Right"
VerticalAlignment="Top" Width="100" Margin="0,136,144,0" Height="30"
Click="prev_vaha_Click" />
```

Výpis 27: Zaregistrování metdoy pro událost Click tlačítka pre_vaha

5. 3 Zhodnocení

Oba Silverlight dynamické jazyky IronPython i IronRuby velice dobře pracují s XAML prezentační vrstvou. Po spojení s aplikační logikou jsou dostupné všechny jednotlivé objektu prezentační vrstvy. Práce v obou jazycích tedy byla velice podobná v obou případech. Za značnou nevýhodu bych ovšem považoval absenci více dobrých materiálů a zdrojů k této integraci dynamických jazyků do Silverlight. Podstatně lépe si v tomto případě vede IronPython, který se jeví jako více podporovaným také ze strany samotného Microsoftu. Jak již bylo zmíněno, tak IronPython se těší také častější a aktuálnější verzi. Právě tohle mohlo být také důvodem, proč se při vývoji ve VS2010 IronPython aplikace jevila pomalejší vůči aplikaci IronPython. Z pohledu přehlednosti by se dal vyzdvihnout IronPython, který je velice čistým jazykem.

6. Výkonostní testy

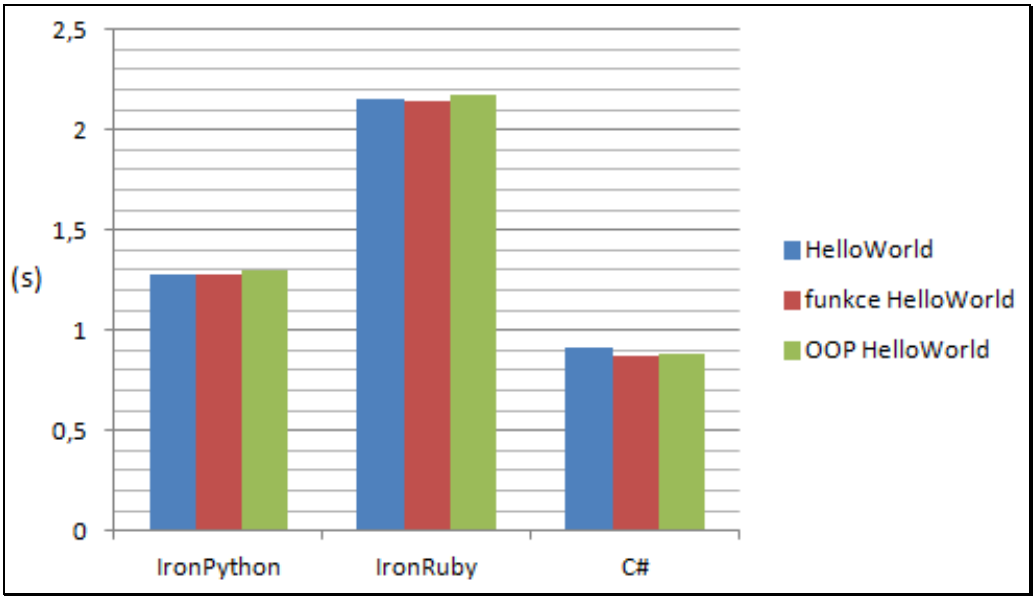
V této kapitole byly provedeny výkonnostní testy dynamických jazyků IronPython, IronRuby a k porovnání také jazyka C#. Všechny zdrojové kódy je možné nalézt na přiloženém DVD.

Výkonnostní testy byly provedeny na počítači:

- Procesor: Intel(R) Core™2 Quad
  - Rychlost: 2.66 GHz
  - Počet jader: 4
- Operační paměť: 2 GB
- Operační systém: Windows 7 Professional SP1 32 bitový

6. 1 Výpis na standardní výstup

Pro příklad otestování výpisu na standardní výstup posloužil příklad, který vypsal hlášku HelloWorld 1000 krát na standardní výstup. Každý jazyk řešil výpisy pomocí příkazu, volání funkce a OOP.

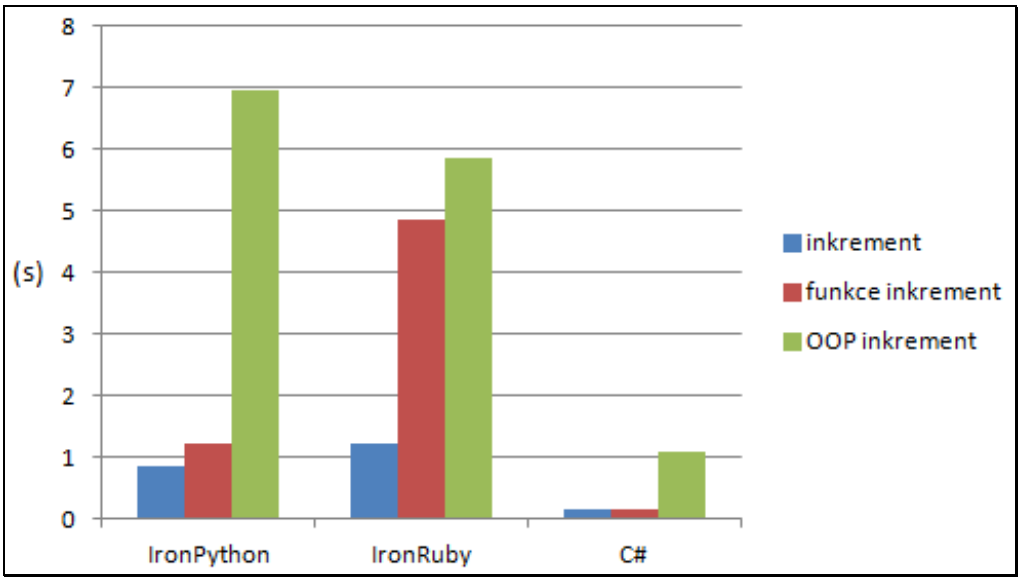


Graf 1: Porovnání průměrné rychlostí(s) jazyků při výpisu na standardní výstup

Protože čím rychlejší, tím lepší, je z grafu 1 je patrné, že nejlepší výsledky na své platformné dosáhl jazyk C#, ovšem o moc pozadu nezůstal IronPython. Nejhorších výsledků dosáhl naopak IronRuby, který byl nejpomalejším. Všechny programy byly spuštěny 10 krát a jejich výsledný čas byl zprůměrován.

6. 2 Inkrementace proměnné

Příklad provedl 10 000 000 krát inkrementaci proměnné. U C# programu se jednalo o inkrementaci proměnné typu dynamic, z důvodu srovnatelnosti inkrementace s ostatními dynamickými jazyky.



Graf 2: Porovnání průměrné rychlostí(s) jazyků při inkrementaci proměnné

Z grafu 2 vychází jako vítěz opět C#. IronPython obstál při inkrementaci přímo v bloku kódu a také pomocí volání metody, ale zde se poprvé v testu poráží IronRuby IronPython a to v příkladu OOP.

6. 3 Fibonacciho posloupnost

V tomto příkladu byla otestována nekonečná posloupnost, kterou navrhl italský matematik Leonardo Pisano známý jako Fibonacci. Jedná se tedy o realizaci Fibbonaciho posloupnosti.

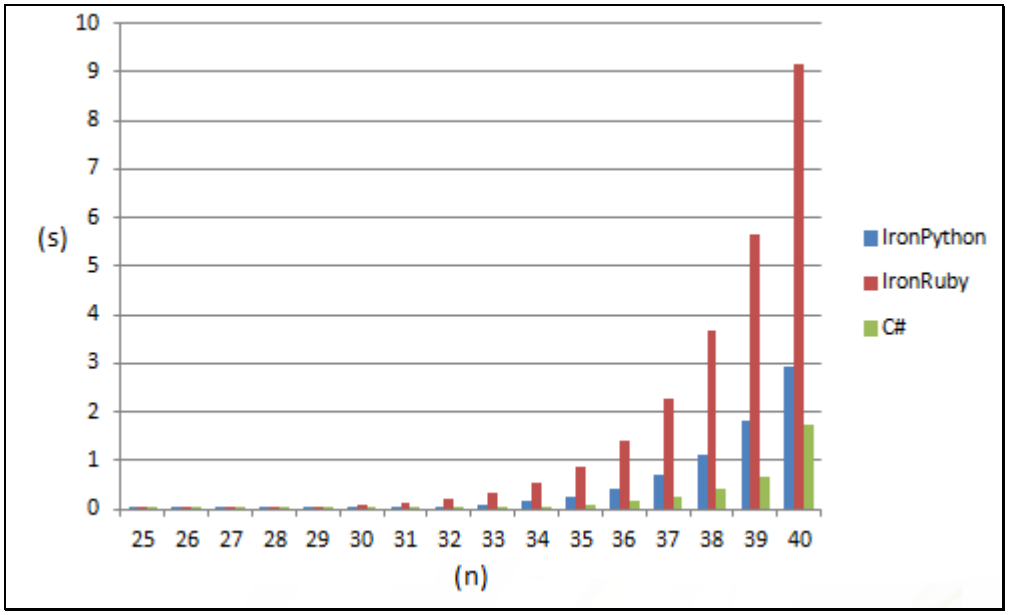
$$F(n) = \begin{cases} 0, & \text{pro } n = 0; \\ 1, & \text{pro } n = 1; \\ F(n-1) + F(n-2) & \text{jinak.} \end{cases}$$

Obrázek 22: Matematická definice Fibonacciho posloupnosti

V kódu vypadá Fibonacciho posloupnost následovně:

```
def fib1(n):
    if n < 2:
        return n
    return fib1(n-1) + fib1(n-2)
```

Výpis 28: Fibonacciho posloupnost zapsaná pomocí IronPython



Graf 3: Porovnání rychlosti(s) Fibonacciho posloupnosti velikosti n

Jak je z grafu 3 patrné, opět z testu nejhůře vyšel IronRuby a to několika násobně než IronPython. Přestože C# pracoval se svým poměrně novým typem dynamic, byl v testu opět nejúspěšnější.

6. 4 Zhodnocení výkonnostních testů

Nejhůře si v testech počínal jazyk ironRuby, který skončil ve většině příkladů poslední. Výrazně lépe dopadl jazyk IronPython, ovšem ani ten nemohl být konkurencí pro C#. Z těchto základních testů tedy lépe vychází C#, ale zdaleka se nejedná o přímý důkaz nepotřebnosti a kompletní nevyužitelnosti těchto jazyků.

Důvodem pro neúspěch IronRuby může být jeho slabší vývoj oproti IronPython. Poslední vydání IronRuby bylo více než pře rokem, oproti poslednímu vydání IronPython, které je v současnosti staré necelé dva měsíce. Kompletně to vypadá, že podpora Microsoftu a hlavně větší komunita táhne IronPython vzhůru mnohem rychleji než IronRuby.

7. Závěr

Cílem práce bylo seznámit se s technologií Silverlight a jejím doplňkem Silverlight Dynamic Languages SDK, tvorba ukázkových příkladů a provedení výkonnostních testů. Při tvorbě této bakalářské práce se bylo nutné seznámit s .NET Frameworkem a také Dynamic Language Runtime. Zjistil jsem, že Silverligh Dynamic Languages nabízejí podobné možnosti jako přirozený jazyk pro Siverlight, kterým je C#. komunita, která vyvíjí dynamické jazyky na platformě .NET udělala kus práce. IronPython si vede celkem obstojně a těší se mnohem většího zájmu, než IronRuby. Na první pohled z hlediska implementace by se mohlo zdát, že jsou si rovni a vývojář by si mohl vybrat podle svého srdce. Není tomu ovšem tak, protože vývoj IronRuby stagnuje již více než rok a půl a není proto možné dostatečně reagovat na novinky v Silverlight. Při tvorbě jsem narazil na velikou propast při přístupu začátečníka k Silverlight Dynamic Languages, tím jsou nedostatečné zdroje a dokumentace, kde v tomhle opět zaostává nejvíce IronRuby, který své souboje naplno prohrál také v provedených výkonnostních testech. Bohužel jak již bylo zmíněno tak Silverlight pomalu umírá díky nástupu HTML5 a tudíž také projekt Silverlight Dynamic Languages umírá s ním. Ovšem všechno není jen kritika a programátoři, kteří vyvíjejí již dlouho v jazycích Python nebo Ruby toto vylepšení uvítají, možnost využití .NET Frameworku se jim zamlouvá a jistě v něm najdou svá pozitiva. Můj názor je ovšem takový, že stále je lepší variantou zůstat u vývoje v jazyce C#, který si právě od .NET verze 4.0, jenž již v sobě zahrnuje Dynamic Language Runtime, může osvojit dynamické vlastnosti a jeho síla se tímto daleko zvětšuje. Pokud ovšem i přesto chcete sáhnout k dynamickým jazykům, je podle mého názoru lepší zvolit IronPython a to z důvodu většího zájmu a vývoje, kterou mu komunita poskytuje, ale také protože IronPython je čistějším a přehlednějším jazykem pro začátečníka.

Seznam obrázků

Seznam výpisu zdrojového kódu:

Seznam tabulek:

Seznam grafů:

Použitá literatura

[1] URL - http://dlr.codeplex.com/documentation (10.4.2012)

[2] MACDONALD, Matthew. *Pro Silverlight 4 in C#* [online]. Updated. Berkeley, Calif.: Apress, 2010 [cit. 2012-05-04]. ISBN 978-143-0229-803.

[3] FOORD, Michael J. *IronPython in action*. Greenwich, Conn.: Manning, 2008, 464 s. ISBN 19-339-8833-9.

[4] URL - http://blogs.msdn.com/b/hugunin/archive/2007/04/30/a-dynamic-language-runtime-dlr.aspx (10.4.2012)

[5] URL – http://www.mono-project.com/Moonlight (10.4.2012)

[6] URL - http://www.ruby-lang.org/en/about/ (26.4.2012)

[7] URL - http://msdn.microsoft.com/en-us/library/w0x726c2.aspx (26.4.2012)

[8] URL - http://cs.wikipedia.org/wiki/Python(26.4.2012)

[9] URL - http://sdlsdk.codeplex.com/wikipage?title=Chiron&ProjectName=sdlsdk (26.4.2012)

[10] URL - http://www.microsoft.com/silverlight/what-is-silverlight/ (26.4.2012)

[11] URL - http://cs.wikipedia.org/wiki/Programovac%C3%AD_jazyk (26.4.2012)

[12] URL - http://cs.wikipedia.org/wiki/Typov%C3%A1_kontrola (26.4.2012)

[14] URL - http://cs.wikipedia.org/wiki/Python (26.4.2012)

[17] URL – http://blogs.msdn.com/b/brunoterkaly/archive/2009/07/23/c-4-0-dynamic-language-runtime.aspx (26.4.2012)

[18] LACKO, Ľuboslav. *Silverlight*: *výukový průvodce tvorbou interaktivních aplikací*. Vyd. 1. Brno: Computer Press, 2010, 464 s. ISBN 978-80-251-2716-2.

[19] URL - http://msdn.microsoft.com/en-us/library/bb822049(v=vs.110).aspx (2.5.2012)

[20] URL - http://channel9.msdn.com/Events/BUILD/BUILD2011/TOOL-834T (2.5.2012)

[21] URL - http://msdn.microsoft.com/en-us/library/dd233052(v=vs.100).aspx (2.5.2012)

[22] URL - http://channel9.msdn.com/blogs/pdc2008/tl10 (2.5.2012)

[23] URL - http://msdn.microsoft.com/cs-cz/library/system.dynamic.dynamicobject.aspx (2.5.2012)

[24] URL - http://msdn.microsoft.com/en-us/library/system.dynamic.expandoobject.aspx (2.5.2012)

[25] URL - http://msdn.microsoft.com/en-us/library/system.dynamic.dynamicmetaobject.aspx (2.5.2012)

[26] URL - http://www.silverlight.net/learn/overview/what's-new-in-silverlight-5 (2.5.2012)

[27] URL - http://www.zive.cz/clanky/internet-explorer-10-umi-html5-ale-bude-to-stacit/sc-3-a-162645/default.aspx (2.5.2012)

Přílohy:

Tato příloha slouží k popsání adresářové struktury DVD přikládaného k této bakalářské práci. Jako kompilátor a vývojové prostředí jsem používal Visual Studio 2010 Professional ve Verzi 10.0.40219.1 s nainstalovaným SP1. Do tohoto vývojového prostředí byly postupně doinstalovány IronPython tools for VS2010 a také IronRuby tool for VS2010. Dále byla doinstalována podpora pro Silverlight 4 a Silverlight 5. Na klientském počítači byl nainstalován .NET Framework 4.

DVD kromě textu bakalářské práce obsahuje také ukázkové příklady a skripty:

Solution – Prevodnik_IP (viz. kapitola 5.2.1)
Solution – Prevodnik_IR (viz. kapitola 5.2.2)
Solutin – Prevodni_CS (viz. kapitola 5.2.3)
Directory – Benchmark – Adresář s použitými skripty (viz. kapitola 6)